

**ИТОГОВЫЙ ПРОЕКТ ПО
ДИСЦИПЛИНЕ
"ОСНОВЫ АЛГОРИТМИЗАЦИИ И
ПРОГРАММИРОВАНИЯ"**

**ТЕМА РАЗРАБОТКА ИГРЫ "КРЕСТИКИ -
НОЛИКИ" НА PYTHON**

РАЗРАБОТАЛИ: СТУДЕНТКИ ИС-24

ЛЮБЕЦКАЯ АНТОНИНА

ПАСИНА ЕЛИЗАВЕТА

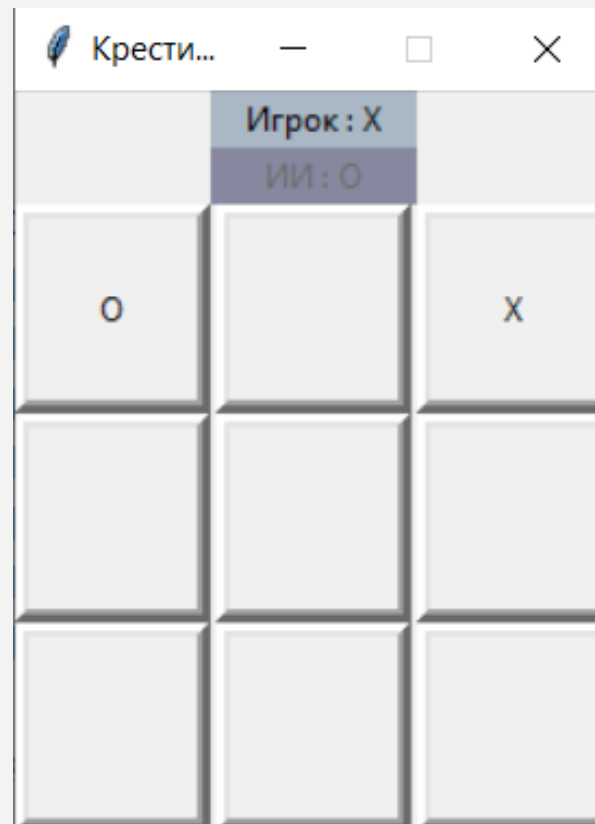
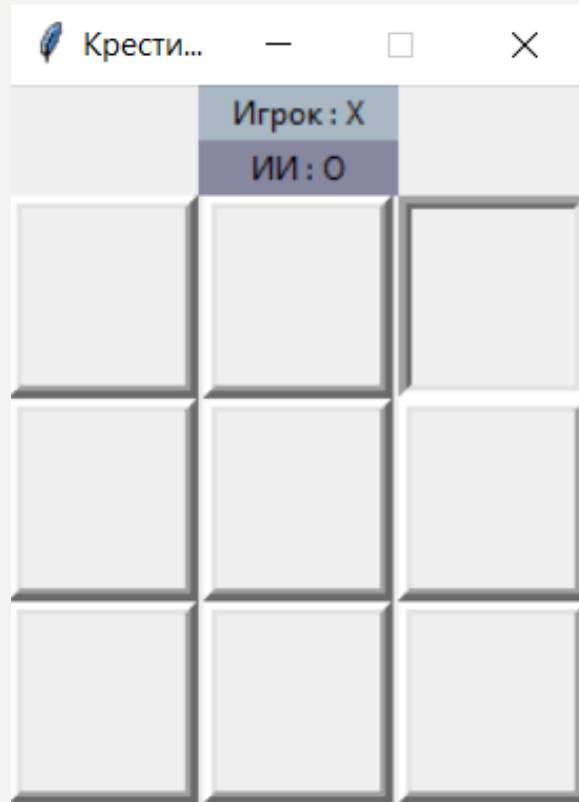
РУКОВОДИТЕЛЬ ПРОЕКТА: МАНАКОВА

ОЛЬГА ПЕТРОВНА

ЗАДАЧИ ПРОЕКТА

- 1) Цель разработки
- 2) Обсуждение дизайна
- 3) Изучение дополнительных библиотек
- 4) Написание программы
- 5) Тестирование и устранение ошибок
- 6) Итоги

ЦЕЛЬ РАЗРАБОТКИ

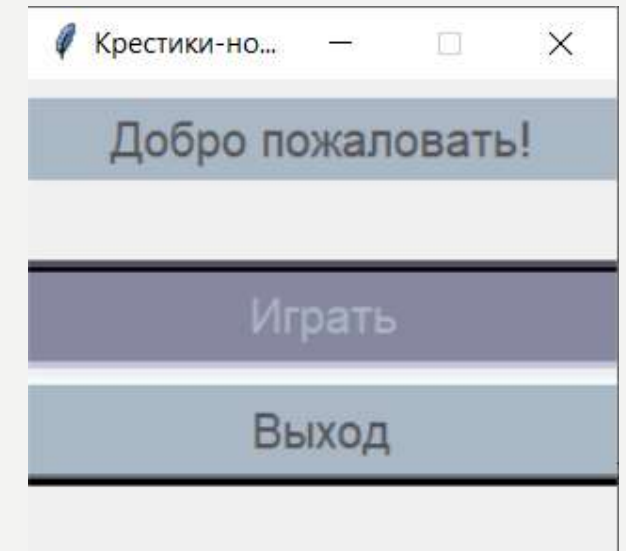
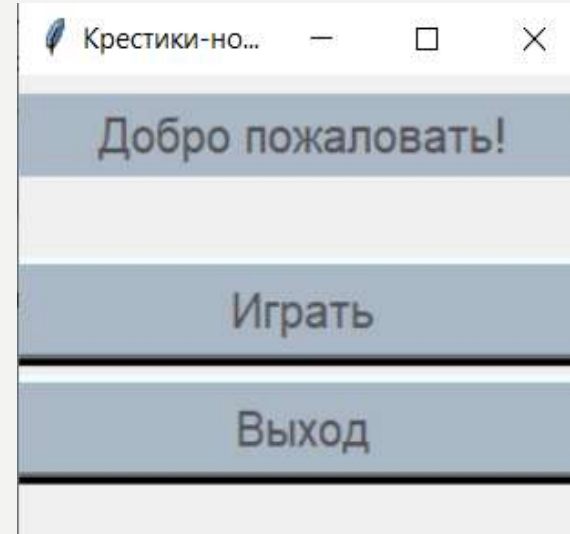


Создать игру
“Крестики-нолики”
на Python с
графическим
дизайном

ОБСУЖДЕНИЕ ДИЗАЙНА

Мы захотели сделать крестики-нолики в минималистичном стиле. Решили использовать светло-голубой цвет для кнопок и для надписи “Добро пожаловать!”. А при нажатии на кнопку будет появляться лиловый цвет.

Мы решили, что наша игра будет не в развернутом виде.



ИЗУЧЕНИЕ ДОПОЛНИТЕЛЬНЫХ БИБЛИОТЕК

Для написания графического интерфейса в Питоне нам понадобятся библиотеки random, functools, copy и tkinter

```
# Импортируем библиотеки. (Рандом нужен для компьютера)  
import random  
  
# Импортируем tkinter  
from tkinter import *  
from functools import partial  
from tkinter import messagebox  
from copy import deepcopy
```

НАПИСАНИЕ ПРОГРАММЫ

```
# создаем пустое пространство  
global board  
board = [" " for x in range(3)] for y in range(3)]
```

Создаем переменную board, которая представляет собой список списков размером 3х3. Каждый элемент списка устанавливает значение " ", что означает пустую ячейку на игровой доске.

НАПИСАНИЕ ПРОГРАММЫ

```
# Проверка победителя матча
# по правилам игры

3 usages
def win(b, l):
    return ((b[0][0] == l and b[0][1] == l and b[0][2] == l) or
            (b[1][0] == l and b[1][1] == l and b[1][2] == l) or
            (b[2][0] == l and b[2][1] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][0] == l and b[2][0] == l) or
            (b[0][1] == l and b[1][1] == l and b[2][1] == l) or
            (b[0][2] == l and b[1][2] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][1] == l and b[2][2] == l) or
            (b[0][2] == l and b[1][1] == l and b[2][0] == l))
```

Функция win(b, l) принимает два аргумента: b - игровую доску и l - символ игрока, в нашем случае "X". Функция проверяет, есть ли на доске выигрышная комбинация для заданного игрока l.

НАПИСАНИЕ ПРОГРАММЫ

Функция `free(i, j)` принимает два аргумента: `i` и `j` - координаты клетки на игровой доске. Функция проверяет, свободна ли данная клетка. Если свободна, то возвращает `True`, если нет - `False`.

А функция `full()` проходит по каждой строке доски и считает количество свободных клеток. Если хотя бы одна клетка свободна, то функция возвращает `False`, что значит игра не закончилась.

```
# Проверяем нажал ли игрок на кнопку
```

```
def free(i, j):  
    return board[i][j] == " "
```

```
# Проверка на заполненность про-ва
```

```
1 usage
```


```
def full():  
    flag = True  
    for i in board:  
        if(i.count(' ') > 0):  
            flag = False  
    return flag
```


НАПИСАНИЕ ПРОГРАММЫ

```
def pc():
    possibl = []
    for i in range(len(board)):
        for j in range(len(board[i])):
            if board[i][j] == ' ':
                possibl.append([i, j])
    move = []
    if possibl == []:
        return
    else:
        for let in ['O', 'X']:
            for i in possibl:
                boardcopy = deepcopy(board)
                boardcopy[i[0]][i[1]] = let
                if win(boardcopy, let):
                    return i
    corner = []
```

Если нет, то функция проверяет, можно ли занять клетки на краях доски (edge). Если можно, то выбирается случайная клетка и возвращаются ее координаты. Если нельзя сделать ход, то функция просто завершается.

Функция pc() отвечает за ход компьютера. Она создает список возможных ходов (possibl). Затем проверяет, есть ли возможность выиграть за один ход. Если есть, то функция возвращает координаты этой клетки. Если нет, то функция проверяет, можно ли занять угловые клетки (corner). Если есть, то выбирается случайная клетка из угловых и возвращаются ее координаты.



```
        for i in possibl:
            if i in [[0, 0], [0, 2], [2, 0], [2, 2]]:
                corner.append(i)
        if len(corner) > 0:
            move = random.randint(0, len(corner)-1)
            return corner[move]
    edge = []
    for i in possibl:
        if i in [[0, 1], [1, 0], [1, 2], [2, 1]]:
            edge.append(i)
    if len(edge) > 0:
        move = random.randint(0, len(edge)-1)
        return edge[move]
```

НАПИСАНИЕ ПРОГРАММЫ

```
def txt_pc(i, j, gb, P1, P2):  
    global hod  
    if board[i][j] == ' ':  
        if hod % 2 == 0: # Процесс определения хода  
            P1.config(state=DISABLED)  
            P2.config(state=ACTIVE)  
            board[i][j] = "X"  
        else:  
            button[i][j].config(state=ACTIVE)  
            P2.config(state=DISABLED)  
            P1.config(state=ACTIVE)  
            board[i][j] = "O"  
        hod += 1  
        button[i][j].config(text=board[i][j])  
    x = True
```

Функция txt_pc() отвечает за ход игрока и компьютера. Она начинается с проверки, свободна ли выбранная клетка. Затем определяет, чей ход - игрока или компьютера. Если это ход игрока, то кнопка, соответствующая выбранной клетке, блокируется, а кнопка другого игрока разблокируется. Затем в board записывается символ 'X' или 'O'. Если это ход компьютера, то вызывается функция pc(). Полученные координаты клетки записываются в board и на соответствующую кнопку выводится символ 'O'. После этого переменная hod увеличивается на 1. В конце функция возвращает значение True.

НАПИСАНИЕ ПРОГРАММЫ

Это продолжение txt_pc . Тут проверяется кто победил. Если победил игрок, то выводится сообщение о его победе и окно игры закрывается. Аналогично, если выиграет компьютер. Если никто не выиграл, то выводится сообщение о ничьей. Если ни одно из этих условий не выполнено, то происходит ход игрока или компьютера в зависимости от того, чей сейчас ход.

```
if win(board, "X"):
    gb.destroy()
    x = False
    box = messagebox.showinfo("Победитель", "Выиграл игрок")
elif win(board, "0"):
    gb.destroy()
    x = False
    box = messagebox.showinfo("Победитель", "Выиграл компьютер")
elif(full()):
    gb.destroy()
    x = False
    box = messagebox.showinfo("Ничья", "Победила дружба")
if(x):
    if hod % 2 != 0:
        move = pc()
        button[move[0]][move[1]].config(state=DISABLED)
        txt_pc(move[0], move[1], gb, P1, P2)
```

НАПИСАНИЕ ПРОГРАММЫ

Создается игровое поле на основе параметров `game_board`, `l1` и `l2`. Параметры `l1` и `l2` используются для отображения игрока и компьютера, соответственно. Функция `gameboard_pc` создает `button` - список кнопок, т.е. клетки игрового поля. Каждая кнопка имеет свои координаты на поле и связана с функцией `txt_pc`, которая будет вызываться при нажатии на кнопку.

```
def gameboard_pc(game_board, l1, l2):  
    game_board.resizable(False, False)  
    global button  
    button = []  
    for i in range(3):  
        m = 3+i  
        button.append(i)  
        button[i] = []  
        for j in range(3):  
            n = j  
            button[i].append(j)  
            get_t = partial(txt_pc, i, j, game_board, l1, l2)  
            button[i][j] = Button(  
                game_board, bd=5, command=get_t, height=4, width=8)  
            button[i][j].grid(row=m, column=n)  
    game_board.mainloop()
```

Для создания кнопок используется цикл `for`. Внешний цикл создает 3 строки кнопок, а внутренний - 3 кнопки в каждой строке. Для каждой кнопки создается функция `get_t` с помощью `partial`, которая передает координаты кнопки и другие параметры в функцию `txt_pc`. Затем создается объект `Button` для каждой клетки игрового поля. Для кнопок прописываем ширину, высоту и команду, которая будет вызываться при нажатии на кнопку. После создания всех кнопок окно запускается с помощью метода `mainloop()`.

НАПИСАНИЕ ПРОГРАММЫ

```
def withpc(game_board):  
    game_board.destroy()  
    game_board = Tk()  
    game_board.title("Крестики-нолики")  
    l1 = Label(game_board, text="Игрок : X", width=10, bg="#aab8c5", activebackground="#aab8c5")  
    l1.grid(row=1, column=1)  
    l2 = Label(game_board, text="ИИ : O", width=10, bg="#8788a0")  
    l2.grid(row=2, column=1)
```

Функция withpc принимает параметр game_board - окно игры.

Сначала функция вызывает метод destroy() для закрытия предыдущего окна игры. Затем создается новое окно с надписью "Крестики-нолики".

Создаются две метки Label: l1 для отображения игрока и l2 для компьютера. Для l1 прописываем белый фон с таким же активным фоном, что и фон окна, а l2 имеет фиолетовый фон.

Обе метки размещаются на игровом поле с помощью метода grid(), который располагает элементы в сетке, заданной строками и столбцами.

НАПИСАНИЕ ПРОГРАММЫ

```
def play():
    menu = Tk()
    menu.geometry("250x250")
    menu.resizable(False, False)
    menu.title("Крестики-нолики")
    wpc = partial(withpc, menu)

    B1 = Button(menu, text="Играть", command=wpc,
                activeforeground='#aab8c5',
                activebackground="#8788a0", bg="#aab8c5",
                fg="#545154", width=500, font='summer', bd=5)

    B2 = Button(menu, text="Выход", command=menu.quit, activeforeground='#aab8c5',
                activebackground="#8788a0", bg="#aab8c5", fg="#545154",
                width=500, font='summer', bd=5)

    B3 = Label(menu, text="Добро пожаловать!",
               activeforeground='#aab8c5',
               activebackground="#8788a0", bg="#aab8c5",
               fg="#545154", width=500, font='summer', bd=5)

    B1.place(relx=0.5, rely=0.4, anchor=CENTER)
    B2.place(relx=0.5, rely=0.6, anchor=CENTER)
    B3.place(relx=0.5, rely=0.1, anchor=CENTER)

    menu.mainloop()
```

Функция play() создает главное меню игры с помощью модуля tkinter. Создаются три кнопки: "Играть", "Выход" и метка "Добро пожаловать!".

Кнопка "Играть" вызывает функцию withpc и передает ей объект главного меню. Кнопка "Выход" вызывает метод quit() для закрытия главного меню. Метка "Добро пожаловать!" просто приветствует нас.

В конце функции вызывается метод mainloop(), который запускает бесконечный цикл обработки событий, пока пользователь не закроет окно.

НАПИСАНИЕ ПРОГРАММЫ

Здесь мы проверяем, является ли данный модуль главным (то есть запускается ли он напрямую, а не импортируется в другой модуль). Если да, то вызывается функция `play()`, которая создает главное меню игры. Если нет, то код в блоке `if` не выполняется

```
# Вызываем начало игры  
if __name__ == '__main__':  
    play()
```

ТЕСТИРОВАНИЕ И УСТРАНЕНИЕ ОШИБОК

На этапе тестирования мы столкнулись с тем, что наше расположение кнопок было не таким как нужно было, поэтому мы изучили и использовали `relx`, `rely` и `anchor`.

`Relx` и `rely` – это относительные координаты (от 0.0 до 1.0) размещения виджета. А `anchor` устанавливает опции растяжения элемента

```
B1.place(relx=0.5, rely=0.4, anchor=CENTER)
B2.place(relx=0.5, rely=0.6, anchor=CENTER)
B3.place(relx=0.5, rely=0.1, anchor=CENTER)
```


ИТОГ

В результате сделанной нами работы, мы смогли написать “Крестики-нолики” с интерфейсом на Python.

Также произвели устранение ошибок и улучшение внешнего вида игры.

ЛИТЕРАТУРА

Лекции по ООП

<https://metanit.com/python/tkinter/2.5.php>

https://translated.turbopages.org/proxy_u/en-ru.ru.7dbe9c6d-64739d37-ff051b03-74722d776562/https/www.geeksforgeeks.org/tic-tac-toe-game-with-gui-using-tkinter-in-python/

<http://adior.ru/index.php/robototekhnika/260-tic-tac-toe>